Proposed Package Object

PDS Management Council Technical Session

November 3, 1999

Joel Wilf

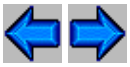
PDS Central Node

Introduction: The Trouble with Software

Including software on a PDS volume presents two potential problems for the data preparer:

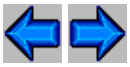
1. Software is platform-specific and may rely on filenames that break ISO9660 compliance (e.g., Makefile).
2. PDS Standards require that every file have a label or be pointed to from a label. And this may be difficult to do for a large collection of source files.

Historically, these problems have been dealt with in unsatisfactory ways, for instance munging filenames or ignoring the labeling requirement, altogether.



Working on the Problem

Steve Joy tackled the problem in an original proposal. This became the starting point for an ad hoc working group that met at the end of September. Another influence was Mark Showalter's work on archiving compressed data. Eric Eliason provided the impetus for finding a solution quickly, since software was going to be included in upcoming MOC volumes. Software delivery is also a key concern for NAIF.



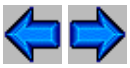
Proposed Solution

The proposed solution is to let the data preparer create a software *package* containing multiple files. The files can be source files, makefiles, binaries, documentation -- anything that can be distributed as part of a single application.

In this scheme, there is only one PDS label, containing the following objects:

- PACKAGED_FILE - which describes the file used as the package container (for example, the .ZIP file containing software files).
- UNPACKAGED_FILE - which describes the actual contents of the package. This object has a sub-object:
  - MANIFEST_TABLE - which describes an index table for the contents of the package.

The PDS label has pointers to these objects, as well as to any files associated with the package but lying outside the PACKAGED_FILE. (For instance, the executable or readme file associated with a package of source code.)

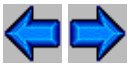


An Example

In this example, there are two versions of the READMOC.EXE program, one for Windows and one for Solaris. The data preparer wishes to make the executable file and a readme file directly accessible on the volume. The source files, however, will be packaged files in a .ZIP file (for Windows) and a .TAR file (for Solaris).

The SOFTWARE directory would look like this:

```
[SOFTWARE]
|
|-SOFTINFO.TXT - Required INFO file for directory
|
|-[PCWIN]
|   |-READMOC.LBL - Label for all READMOC.* files in this directory
|   |-READMOC.EXE - Windows executable
|   |-READMOC.ZIP - ZIP package of Windows source files
|   |-READMOC.TXT - Description of ZIP package
|   |-READMOC.TAB - Table of the contents of the ZIP package
|
|-[SOLARIS]
|   |
|   |-READMOC.LBL - Label for all READMOC.* files in this directory
|   |-READMOC.EXE - Solaris executable
|   |-READMOC.TAR - TAR package of Solaris source files
|   |-READMOC.TXT - Description of TAR package
|   |-READMOC.TAB - Table of the contents of the TAR package
```

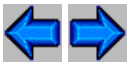


PDS Label for the Example

READMOC.LBL for the PCWIN directory would look like this:

```
PDS_VERSION_ID               = PDS3
LABEL_REVISION_NOTE          = "E. Eliason, 1999-03-17"
DATA_SET_ID                  = "SOME-DATA-SET"
PRODUCT_ID                   = "READMOC"
RECORD_TYPE                  = UNDEFINED
INTERCHANGE_FORMAT           = BINARY
PRODUCT_CREATION_TIME        = 1999-03-16
NOTE                         = "
If you receive this software source code in a zipfile, then
you must extract the files using the ‘‘unzip’’ command. There
are numerous versions of the Zip/Unzip utilities. If you have
one installed already, refer to its documentation in order to
unzip these files. In the case that there is no unzip utility
installed, or it does not unzip correctly, there is a version
of InfoZip at the PDS web site (http://www.pds_site_here.gov)."
SOFTWARE_NAME                = "READMOC"
SOFTWARE_VERSION_ID          = "2.17"
SOFTWARE_LICENSE_TYPE        = "PUBLIC DOMAIN"
TECHNICAL_SUPPORT_TYPE       = "NONE"
PLATFORM                     = "WINDOWS NT"

^PACKAGED_FILE               = "READMOC.ZIP"
^EXECUTABLE_SOFTWARE         = "READMOC.EXE"

  OBJECT                     = PACKAGED_FILE
   ENCODING_TYPE             = ZIP
   REQUIRED_STORAGE_BYTES    = 47751
  END_OBJECT                 = PACKAGED_FILE

  OBJECT                     = UNPACKAGED_FILE
   ^DESCRIPTION              = "READMOC.TXT"
   ^MANIFEST_TABLE           = "READMOC.TAB"

   OBJECT                    = MANIFEST_TABLE
    INTERCHANGE_FORMAT       = ASCII
    ROWS                     = 5
    COLUMNS                  = 4
    ROW_BYTES                = 79
    DESCRIPTION              = "Manifest for package"

    OBJECT                   = COLUMN
     NAME                    = PATH_NAME
     DATA_TYPE               = CHARACTER
     START_BYTE              = 2
     BYTES                   = 32
     FORMAT                  = "A32"
     DESCRIPTION             = "Path to the given file, in UNIX format
                                (slashes separate directory names)"
    END_OBJECT               = COLUMN

    OBJECT                   = COLUMN
     NAME                    = FILE_NAME
     DATA_TYPE               = CHARACTER
```

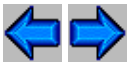
```
     START_BYTE              = 37
     BYTES                   = 12
     FORMAT                  = "A12"
     DESCRIPTION             = "Name of the data file, in upper-case,
                               with extension"
    END_OBJECT               = COLUMN

    OBJECT                   = COLUMN
     NAME                    = BYTES
     DATA_TYPE               = CHARACTER
     START_BYTE              = 52
     BYTES                   = 7
     FORMAT                  = "A7"
     DESCRIPTION             = "Number of bytes in file"
    END_OBJECT               = COLUMN

    OBJECT                   = COLUMN
     NAME                    = CREATION_DATE
     DATA_TYPE               = CHARACTER
     START_BYTE              = 62
     BYTES                   = 15
     FORMAT                  = "A15"
     DESCRIPTION             = "File creation date:  17-Mar-99 10:45"
    END_OBJECT               = COLUMN
   END_OBJECT                = MANIFEST_TABLE
  END_OBJECT                 = UNPACKAGED_FILE
END
```



Index Table for the Example

Assume that there are five source files: Makefile, main.c, main.h, display.c, and display.h, which when unzipped will reside in the READMOC/src directory. Then READMOC.TAB would look like this:

```
"READMOC/src            ","Makefile    ","   1275","17-Mar-99 10:49"
"READMOC/src            ","main.h      ","   6485","17-Mar-99 10:49"
"READMOC/src            ","main.c      ","  25872","17-Mar-99 10:49"
"READMOC/src            ","display.h   ","   1656","17-Mar-99 10:50"
"READMOC/src            ","display.c   ","  12463","17-Mar-99 10:48"
```



Some Issues

- Should we require that certain files (e.g., executables or description files) always be available outside of the package?
- Should we require that certain files (e.g., executables) appear inside the package?
- Which formats for packages should be allowed? (Currently, only .ZIP has been approved. If the criteria is that source code for packaging/unpackaging be available, then .TAR and .GZ would qualify, since the source is available from the Free Software Foundation.)
- Should a plain directory tree be allowed as a packaging format?
- Should the data preparer be allowed to use a plain-text manifest instead of an index table?

Summary

Issues remain to be worked out. However, the proposed solution does address the problems mentioned at the beginning of this talk:

1. It solves the file-naming problem, since files that only appear inside the package don’t have to conform to ISO9660.
2. It solves the every-file-needs-a-label problem, since only a single label is required.